Deep Learning COSC 2779 Assignment 1   Syed Haider Saleem (s3796258)

Problem Overview

The purpose of this report is to present the problem of image classification that is to estimate Head pose of a person. To estimate the head pose we need to predict two labels that are: Tilt - Vertical angle of the head, Pan - Horizontal angle of the head. Since we need to be predicting two labels at a time hence this type of problem comes under the umbrella of multilabel classification and hence we will treat it accordingly.

Data Set

The dataset provided to us for this problem was taken from Head Pose Image Database. The dataset consists of a total of about 2790 numbers of 192x144 images. The dataset was then further divided into a training and test set where the training set consists of 1860 images where each image contains a tilt and pan angle, test set consists of about 465 images whose labels we need to predict using our model. The data distribution across various classes seems fairly equal hence we won't be needing to tackle this fairly common issue.

Data Preprocessing

Data preprocessing more often than not is the most time consuming and important phase of the project and it was no exception this time. I tried a fair bit of techniques to preprocess our images that includes image normalization, resizing images, label binarizing and most importantly data augmentation.

- Data Augmentation

  This was probably the most important step of this project, as before augmentation I was getting an extremely poor accuracy score even below 50% overall, classification of Pan - Horizontal angle was specifically poor since we had a small data set. I augmented images in five different ways: by changing brightness, RGB shift, RandomGamma and Hue Saturation value and I did it for each image hence after augmentation I was left with five times more training data than before. To make sure we are not creating biasness I made sure we only augmented training images so that test data might not get tempered. This technique gave a huge boost to my validation accuracy.

- Image Resizing

  Original images were of 192x144 size, I resized them to 144x144 to make the height and width same across all the images, it also helps in making our model train faster on such size.

- Data Normalization

  Images were also normalized. Normalization helps in converging the model faster as it rescales the pixels values and brings in range of [0-1].

- Cleaning Data

  Apart from these major preprocessing steps we also had to clean our data that includes renaming a column and dropping a few unnecessary columns like series and id.

- Label Binarizer

  Since our problem is multilabel classification, to feed the multi label data to our CNN model we had to label binarize our y_train and y_val dataset. After applying the transform function of the label binarizer we got 9 columns (classes) of tilt and 13 columns (classes) of pan which is the number of classes in each label. We also had to apply inverse_transform at the end to retrieve our labels.

- Training and Validate split

  After preprocessing the data we are then going to split our data into training and validation sets. Training set consists of about 80% of data while validation consists of only 20%. This validation dataset is going to be used as a baseline data to determine which model we will choose.

Model Selection and Approach

Choosing a model is fairly a complex process when you are dealing with multilabel classification. For this purpose I choose to experiment with quite a few types of Convolutional Neural Networks based on some of the research/ literature work available. One of the research work that I was quite fascinated with was the Branching CNN method where branches of neural networks are trained to predict a specific label and since we had two labels I thought why not try it out and create a BCNN. I spent quite a bit of time on it to finally realise that Pan - Horizontal angle branch is not really able to predict pan with good accuracy hence I have to leave it and go for conventional neural networks. I will still talk about both of these approaches. An important thing to note is that we will be treating this problem as a classification problem since we have fixed a finite number of classes to predict from in each label. Also it is important to mention that since we are dealing with multi label classification what our basic approach in this case was to label binarize each label i.e tilt and pan and get 9 classes (columns) for tilt and 13 classes (columns) for pan and then we combine all these classes (columns) to make our Ylabel a dataframe of 22 columns which are either 0 or 1 and we predict classes.

Evaluation Metric

Before diving in the details of both models, let's talk about the evaluation metric I used to evaluate my model. As said before the evaluation was done on validation data set and **"Accuracy"** was used as our evaluation metrics since the problem type is fairly straightforward, we need to predict correct labels and the distribution of each class in each label is almost equal, accuracy would be a good metric.

B-CNN: Branch Convolutional Neural Network (First technique)

After reading a research paper (Branch Convolutional Neural Network for Hierarchical Classification) published on this approach, I tried applying a similar BCNN on my dataset. The amazing thing about this network is that you can create branches where each branch could then be used to predict a specific label which in our case is tilt and pan. Each branch could have its own loss function and accuracy metrics and layers corresponding to each branch could be customised as well to tackle the problem. This is specifically critical where some labels are harder to predict than others, in our case Pan was harder to predict hence I opted to go for it in the start. Its structure consists of the first layer of Convolution having a 3x3 filter with non linear activation of 'RELU', batch normalization and max pooling. Then flatten were applied and a dense of 32 were applied, dropout of 0.5 was also used and then the last layer consisted of output of 9 for our first branch as tilt had 9 unique classes to predict from. Another such layer was applied on the second branch to predict pan label that had 13 output nodes to predict one of its 13 classes. For optimizer 'adam' was used and binary_crossentropy along with accuracy was used for both labels. Accuracy of about 80% was achieved for tilt label while only an accuracy of 20% was achieved on pan label. Since it did not perform as expected hence I had to move to another model which is sequential and hence I won't discuss this in detail.

Convolutional Neural Network - sequential (Final model used)

A sequential **Convolutional Neural Network (ConvNet/CNN)** is then applied to solve the problem. CNN generally performed well in image classification mainly as it has the ability to learn and extract features from the raw image data.

Model Design(CNN)

So as always we started with the simplest structure of neural network with one layer of convolution, batch normalisation and max pooling (without any regularisation) and then we started building on it by adding more layers once we realised that training accuracy is not increasing. Our input shape to the model was 144x144 image and the output contained 22 nodes where each output node corresponds to a class (9 incase of tilt and 13 incase of pan). Our final model contained four layers (complex enough to deal with the problem), at each layer we applied a 3*3 convolutional kernel to achieve a feature map on which we then apply a non linear activation function (relu) to solve nonlinear problems, we also use batch normalization and max pooling on top of that. We also incorporated regularisation using kernel_regularizer and drop out, to reduce overfitting and increasing robustness of our model. After these four layers we use flatten to convert our 3D features to 1D feature vectors and then we create a fully connected layer by using Dense and apply non linear activation again followed by a drop out of 0.5 (this is also repeated twice) and then a final non activation function of sigmoid is used as our label classes are either zero or one and sigmoid performs best when predicting between 0 and 1. We are going to use ‘adam’ as our optimizer as it performs well in case of image classification (learned from lecture slides), for loss function we are using ‘binary_crossentropy’ and as discussed before our evaluation metrics is ‘accuracy’.

Important features of this model

- Each Layer - Conv2D, BatchNormalization, MaxPooling2D, Dropout
- Consist of 4 of these layers
- Optimizer: adam, loss: binary_crossentropy, metrics: accuracy

Model Training

Model was then trained on training data for 30 epochs with early stopping.

Model Evaluation (Results)

CNN gave an overall validation accuracy of 76% overall (30 epochs).

Ultimate Judgement and Limitations:

Both techniques (Branch CNN and sequential CNN) in my opinion had their own strengths and weaknesses yet BCNN performed poorly on pan label hence we had to settle for sequential CNN. Also I don't believe this model could be deployed and used practically considering the comparatively low accuracy of 76% hence we cannot rely on it. To improve the model in the future I believe we need to have a bigger dataset with more variance (as exists in real life) .Also the current dataset was carefully made under experimental conditions and since the real life data have much more variance hence our model might not perform well on real life images. Also in real life the angles for tilt and pan might not lie in the specified range and the images might have different backgrounds, there might be distortion or the picture might not have been taken at a good angle. Yet still for this particular data we did get reasonable accuracy specifically for tilt. For a pan label we might need to get more data to make more accurate predictions on it.

Appendices

Branch Convolutional Neural Network

Model Type	**Tilt Accuracy (validation)**	**Pan Accuracy (validation)**
BCNN	0.8108	0.1935

Table 1.1

Sequential CNN

Model Type	**Overall Accuracy (validation)**
CNN	**0.7656**

Table 1.2

Table 1 represents the accuracy achieved by using different models

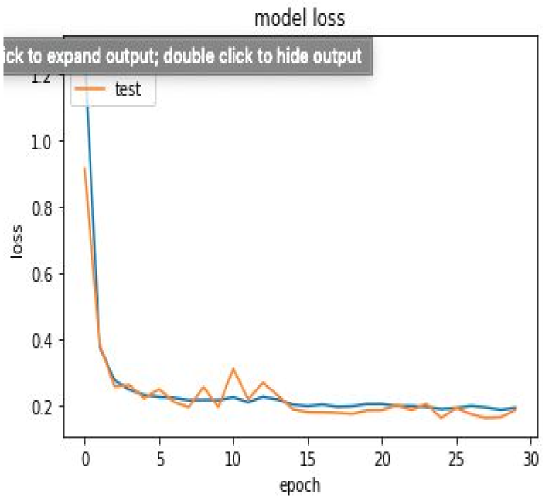

Fig 3.1 (Loss function)

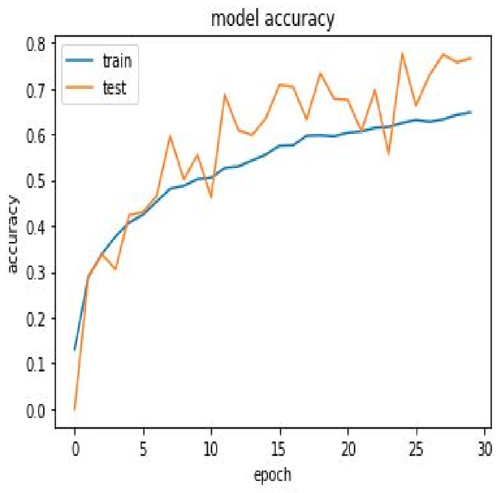

Fig 3.2 (Accuracy)

Figure 3 shows the loss and accuracy (results) of our trained model

References:

[1]"B-CNN: Branch Convolutional Neural Network for Hierarchical Classification", GroundAI, 2020. [Online]. Available: https://www.groundai.com/project/b-cnn-branch-convolutional-neural-network-for-hierarchical-classification/2. [Accessed: 07- Sep- 2020].

[2]"Multi-Label Image Classification with Neural Network | Keras", Medium, 2020. [Online]. Available: https://towardsdatascience.com/multi-label-image-classification-with-neural-network-keras-ddc1ab1afede. [Accessed: 07- Sep- 2020].